Read More, Think More：
Web エージェントにおける観測軽量化の再検証

榎本昌文　小原涼馬　張皓辰　小山田昌史
NEC データサイエンスラボラトリー
{masafumi-enomoto,ryoma-obara,haochen-zhang,oyamada}@nec.com

概要

大規模言語モデルを用いて Web 操作を行うエージェントにおいて，HTML に代表される Web ページの観測は行動選択の基盤となる．既存研究では HTML の長大さが性能を妨げるとされ，軽量化がトレンドとなってきた．本研究ではこのトレンドを再検証し，以下を明らかにした：(1) オープンソースモデルでは簡素な観測（アクセシビリティツリー）が有効だが，高性能な商用モデルではむしろ詳細な観測（HTML）が概して有効である (2) 商用モデルは HTML のレイアウト情報を活用できる一方，オープンソースモデルでは入力増大によりハルシネーションが増加する (3) 推論時の思考トークン量を増加させることで，詳細な観測の効果を高めることができる (4) 観測履歴の追加は多くのモデル・設定で有効であり，差分表現による効率化も可能である．

1　はじめに

大規模言語モデル（LLM）を用いて Web 操作を行うエージェント（Web エージェント）は，フォーム入力など反復的なタスクを自動化できることから，近年活発に研究されている [1]． 一般に，Web エージェントは行動の実行と観測の獲得のループ [2] で実装される：エージェントは各ステップにおいてクリックなどの行動を実行し，その結果として現在の Web ページを観測する． Web エージェントはこのループを，タスクを終了したと判定するまで実行する． Web ページの観測に含まれる情報は，実行可能な行動の把握やタスク進捗の理解において重要である．

既存の Web エージェント研究では，観測情報の軽量化が一貫したトレンドとなってきた． その主な理由は，Web ページの文書構造を直接テキスト化した HTML が長大だからである． 例えば，Web エージェントのベンチマークである WorkArena では，HTML は最小 4 万〜最大 50 万トークンの範囲に分布している [3]． 初期の LLM では入力長制約により，これを全て入力できなかった [4]． 現在では Gemini モデル [1] に代表される 100 万トークン規模の入力長を持つ LLM が登場しており， モデルによっては HTML をそのまま入力可能である． しかし，タスクに関連しない情報が推論を妨げる [5] という理由から，軽量化は依然として主流となっている． 具体的には，タスクに関連する要素のみの抽出 [4, 6, 7]，簡素な表現への変換 [5, 8]，より軽量なアクセシビリティツリー（a11y）の利用 [9, 10] など，様々な軽量化のアプローチが提案されてきた．最新の観測に加えて過去ステップの観測履歴を活用する場合も，同様に軽量化が行われている [5, 11].

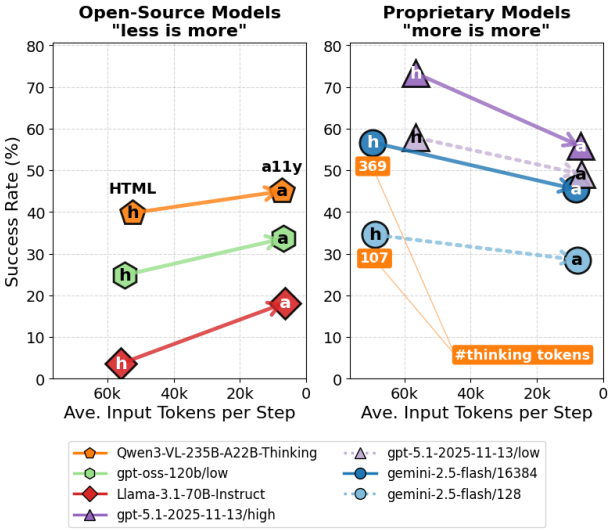

図 1　観測の表現形式とタスク成功率の関係（WorkArena L1）．x 軸は各ステップの平均入力トークン数，y 軸はタスク成功率．h は HTML，a はアクセシビリティツリー（a11y）を表す．矢印は同一モデルにおける HTML から a11y への変更を示す．左図：オープンソースモデルでは観測の軽量化（h -> a）により成功率が向上する．右図：高性能な商用モデルでは逆に成功率が低下する

1）　https://gemini.google.com/?hl=ja

より詳細な関連研究については付録 A.1 を参照されたい．

本研究では，この観測軽量化のトレンドを再検証する．従来，軽量化が有効とされてきた背景には，LLM の長文処理能力の限界があった．しかし，近年の LLM は長大な入力の処理に長けており [12]，推論時に多くの思考トークンを生成することで，さらに処理能力を向上させるという報告もある [13]．こうした進展を踏まえて，本研究では最新の LLM を用いて**「観測情報の軽量化の効果は，モデル能力や思考トークン量によってどのように変化するか？」**というリサーチクエスチョンを検証した．検証の結果，以下が明らかになった：**オープンソースモデルのような低性能なモデルでは簡素な観測（a11y）が有効だが，商用モデルのような高性能なモデルでは逆に詳細な観測（HTML）が有効である**（図 1，§3.2）．また，思考トークン量を増やすことで，長大な観測の効果を高めることができる．行動の分析から，高性能モデルは HTML の詳細なレイアウト情報を活用して適切な行動を選択できる一方，低性能モデルでは入力の増大によりハルシネーションが増加することが示唆された（§3.3）．さらに，観測履歴の追加は多くのモデル・設定で有効であり，差分表現を用いることで効率化できることも分かった（§3.4）．

2 問題設定

先行研究 [5] に倣い，我々は Web エージェントタスクを部分観測マルコフ決定過程（POMDP）として形式化する：$\langle S, A, O, P, R, \Omega \rangle$．ここで，$S$ は状態空間，A は行動空間，O は観測空間，$P : S \times A \to S$ は状態遷移関数，$R : S \times A \times S \to \mathbb{R}$ は報酬関数，$\Omega : S \to O$ は観測関数である．

LLM ベースのエージェントは以下のように動作する．各ステップ t において，エージェントは方策 $\pi_{\text{LLM}}(a_t | I, o_t, (o_i, a_i)_{i=0}^{t-1})$ に従って行動 $a_t \in A$ を選択する．ここで I はタスク指示を含むシステムプロンプト，o_t は現在の観測，$(o_i, a_i)_{i=0}^{t-1}$ は過去の観測-行動ペアの履歴である．行動 a_t が実行されると，環境は状態遷移関数 P に従って次の状態 $s_{t+1} = P(s_t, a_t)$ へ遷移し，エージェントは観測関数 Ω を通じて新たな観測 $o_{t+1} = \Omega(s_{t+1})$ を受け取る．観測 o_{t+1} は Web ページをテキスト化した表現（HTML，a11y）やレンダリングした画像（スクリーンショット）として与えられる．このプロセスは，

表 1 各観測形式・モデルにおけるタスク成功率（%）．WorkArena L1 で評価．スコア横の括弧内に a11y からの差分を示す．N/A は当該モデルが画像入力に非対応であることを示す．モデル名末尾の T は Thinking, I は Instruct を示す．モデル名の括弧内の high/low は `reasoning_effort`，数値は `thinking_budget` を示す．入力トークン数は参考値として gpt-5.1 (high) のものを掲載（画像入力も含む）．

	a11y	html	a11y+scrs
入力トークン数	6720	56653	7446
商用モデル			
gpt-5.1-2025-11-13 (high)	55.8	73.3 (+17.5)	59.7 (+3.9)
gpt-5.1-2025-11-13 (low)	49.1	57.9 (+8.8)	55.8 (+6.7)
o3-mini-2025-01-31 (high)	39.7	32.1 (-7.6)	N/A
gemini-2.5-flash (16384)	45.5	56.7 (+11.2)	48.5 (+3.0)
gemini-2.5-flash (128)	28.5	34.5 (+6.0)	32.7 (+4.2)
オープンソースモデル			
gpt-oss-120b (high)	46.7	38.8 (-7.9)	N/A
gpt-oss-120b (low)	33.6	24.8 (-8.8)	N/A
gpt-oss-20b (high)	46.4	27.6 (-18.8)	N/A
gpt-oss-20b (low)	20.0	18.2 (-1.8)	N/A
Qwen3-VL-235B-A22B-T	45.0	39.8 (-5.2)	44.8 (-0.2)
Qwen3-VL-30B-A3B-T	33.2	26.1 (-7.1)	33.9 (+0.7)
Qwen3-VL-30B-A3B-I	22.7	25.5 (+2.8)	24.8 (+2.1)
Llama-3.1-70B-I	18.2	3.6 (-14.6)	N/A
Llama-3.1-8B-I	0.0	1.2 (+1.2)	N/A

エージェントが終了行動を実行するか，最大ステップ数 $T_{\max}$ に達するまで繰り返される．タスクの成否は，エピソード終了時に報酬 $r = R(s_T, a_T, s_{T+1})$ によって評価される（T は終了ステップ）．

本研究では，観測を方策 π_{LLM} に入力する際の設計として，以下の二つを変化させる：(1) **現在の観測の表現形式**: 観測 o_t を HTML 形式，a11y 形式，スクリーンショット，またはこれらの組み合わせで表現．(2) **観測履歴の長さ**: 履歴として含める過去の観測 $\{o_i\}_{i<t}$ の数．これらの観測設計と言語モデル・思考トークン量の組み合わせを評価する．

3 実験

3.1 実験設定

ベンチマーク 我々は Web エージェントベンチマークである **WorkArena** [3, 14] を評価に用いた．WorkArena は ServiceNow プラットフォーム[2] を用いて，物品の注文やフォーム入力といった日常業務の遂行能力を評価するベンチマークである．本実験では，33 種類のタスクタイプ ×10 シードで計 330 タスクから構成される WorkArena L1 を用いた．

言語モデル 観測設計の効果を評価するため，以下の観点で多様なモデル構成を用意した：(1) モデル能

2） https://www.servicenow.com/jp/

力については，商用モデルとオープンソースモデルの比較，同一ファミリ内でのモデルサイズによる比較（例：gpt-oss-120b/20b）を行った．(2) 思考トークン量については，各推論モデルにおいて制御パラメータを変化させて評価した．商用モデルについては，gpt-5.1-2025-11-13[3]，o3-mini-2025-01-31[4]，gemini-2.5-flash [15] を用いた．オープンソースモデルについては，gpt-oss-120b/20b [16]，Qwen3-VL ファミリ（235B-A22B-Thinking，30B-A3B-Thinking，30B-A3B-Instruct）[17]，Llama-3.1 ファミリ（70B-Instruct，8B-Instruct）[18] を用いた．gpt-5.1，o3-mini，gpt-oss では reasoning_effort パラメータ（low/high）により，gemini では thinking_budget パラメータ（128/16384）により思考トークン量を制御した． Llama-3.1 ファミリは思考トークンを生成しない旧世代のモデルでの性能を評価するために用いた．

行動のグラウンディング エージェントが出力した行動文字列を実際の Web 操作に紐付ける処理をグラウンディングと呼ぶ．本実験では，id ベースのグラウンディングを採用する．具体的には，エージェントは最新の観測 o_t に含まれる DOM 要素の id を指定することで行動 a_t を実行する．行動 a_t は，行動種別，操作対象の id，および必要に応じてパラメータを組み合わせた文字列（例：fill('12', 'hello')）として表現される．出力された行動は，ブラウザ自動操作パッケージである Playwright[5] を通じて実行される．実験フレームワークには AgentLab [19, 3] を用いた．最大ステップ数 $T_{\max}$ はフレームワークのデフォルト設定に従い 15 とし，超過時はタスクを失敗として扱う．

3.2 観測の表現形式とタスク成功率

本節では，観測の表現形式がタスク成功率に与える影響を検証する． a11y は Web ページからインタラクティブ要素を中心に抽出した簡潔な表現である[6]． 一方，HTML は Web ページのより詳細な文書構造を保持しており，本実験の設定ではさらに CSS に基づくレイアウト情報も含めている．a11y を基準として，(1) HTML に置き換えた場合，(2) スクリーンショットを追加した場合（a11y+scrs），の二つを検証した．表 1 に結果を示す．

3） https://openai.com/ja-JP/index/gpt-5-1/
4） https://openai.com/ja-JP/index/openai-o3-mini/
5） https://playwright.dev/
6） グラウンディングのために，DOM 要素 id は保持する．

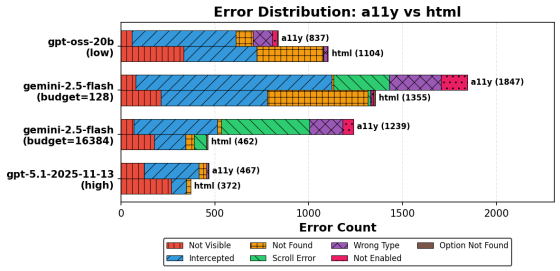

図 2 a11y から HTML への変更に伴うグラウンディングエラー数の増減とその内訳．

モデル能力と観測表現の関係 結果から，高性能なモデルほど情報量の多い観測を活用できることが分かる． SOTA の商用モデル（gpt-5.1, gemini-2.5-flash）では a11y より HTML のほうが性能が良い一方，オープンソースモデルでは HTML を用いると性能が劣化するケースが多い． a11y にスクリーンショットを併用した場合でも，同様に商用モデルのほうが効果が大きい． 同一ファミリ内での異なるモデルサイズを比較すると，モデルサイズが小さいときのほうが HTML による性能の劣化が大きい傾向がある． 例えば，reasoning_effort 設定が同じ high のとき， gpt-oss-120b/20b を比較すると，後者のほうが影響が大きい．また，商用モデルであっても o3-mini のような旧来の小型推論モデルでは，gpt-5.1 と比べると HTML の効果が限定的である．

思考トークン量と観測表現の関係 思考トークン量が多いほど，詳細な観測である HTML を活用できることが分かる． 例えば，同じ gpt-5.1 でも，reasoning_effort を low から high にすることで，HTML による性能向上が大きくなる．また，思考トークンを生成しない Llama-3.1-70B では，HTML の使用により大幅に性能が低下しており，この傾向と整合的である．

3.3 グラウンディングエラー分析

実験設定 §3.2 の結果をより詳細に分析するため，本節ではエージェントが各ステップで出力した行動が環境で正しく実行されたか否か（グラウンディングの成否）を検証する．Playwright による行動実行時に例外が発生した行動をグラウンディングエラーとしてカウントし，エラーの文字列パターンに基づいて分類した．a11y から HTML への変更で成功率が向上したモデル（gemini-2.5-flash，gpt-5.1(high)）と低下したモデル（gpt-oss-20b(low)）を比較分析した．エラー分類の詳細な説明については，付録 A.2 を参照されたい．

エラーの内訳 図 2 に，a11y から HTML への変更に伴うエラー数の増減とその内訳を示す．商用モデルの `gemini-2.5-flash`，`gpt-5.1(high)` ではエラー総数が減少する一方，オープンソースモデルの `gpt-oss-20b(low)` では増加している．このことから，HTML への変更がグラウンディングの成否に影響を与え，それがタスク成功率の差につながっていると考えられる．以下，エラー種別ごとに分析する．

not found エラーは，DOM 内に存在しない要素の id を行動の対象として指定してしまうエラーである．このエラーは `gpt-oss-20b(low)` では大幅に増加する一方，`gemini-2.5-flash(budget=16384)` では微増にとどまっている．この結果は，商用モデルのほうが長文読解能力に優れており，存在しない id を参照する挙動（ハルシネーション）を抑制できていることを示唆する． また，同じ gemini モデルでも `gemini-2.5-flash(budget=128)` では大きく増加していることから，思考トークン量の増加もハルシネーションの抑制に寄与することが分かる．

Intercepted エラーは，操作対象の要素の上に別の要素が重なっているときに操作が奪われてしまうエラーである．このエラーは全てのモデルで減少しており，特に商用モデルでは顕著である．本実験で用いた HTML には，要素の前後関係を指定する z-index を含む CSS スタイル情報が含まれている．商用モデルはこの情報から要素の重なりを把握し，エラーを回避できたと考えられる．

まとめ 以上から，商用モデルでは HTML の豊富なレイアウト情報を活用してグラウンディングエラーを減らせる一方，オープンソースモデルでは情報過多によるハルシネーションでエラーが増加することが分かった．

3.4 観測履歴とタスク成功率

本節では，過去の観測履歴を用いることの効果を検証する． 観測の表現形式は a11y に固定し，履歴なし（hist0），過去 4/9 ステップ分の履歴あり（hist4/hist9）を比較した． また hist9 については，完全な観測を追加する方法（full）と，一つ前の観測との文字列的な差分のみを追加する方法（diff）の二つを検証した． 表 2 に実験結果を示す．

観測履歴の効果 実験結果から，ほとんどのモデル・設定で観測履歴の追加は性能を向上させることが分かる． さらに，`o3-mini` や `gemini-2.5-flash` では

表 2 各観測履歴設定・モデルにおけるタスク成功率（%）．WorkArena L1 で評価．スコア横の括弧内に hist0 からの差分を示す．hist0 は履歴なし，hist4/hist9 は直前 4/9 ステップの観測履歴を追加．hist9 列の full は完全な観測履歴，diff は前ステップとの差分のみを追加した設定を示す．モデル名末尾の/H と/L は `reasoning_effort` の high/low，または `thinking_budget` の 16384/128 を示す．入力トークン数は参考値として gpt-5.1(high) のものを掲載．

	hist0	hist4	hist9 full/diff
入力トークン	6720	26184	39011 / 13670
商用モデル			
gpt-5.1/H	55.8	58.8 (+3.0)	58.8 (+3.0) / 58.8 (+3.0)
gpt-5.1/L	49.1	53.0 (+3.9)	50.9 (+1.8) / 53.3 (+4.2)
o3-mini/H	39.7	43.0 (+3.3)	43.3 (+3.6) / 46.1 (+6.4)
gemini-2.5-f/H	45.5	48.2 (+2.7)	50.0 (+4.5) / 48.2 (+2.7)
gemini-2.5-f/L	28.5	39.4 (+10.9)	39.4 (+10.9) / 33.3 (+4.8)
オープンソースモデル			
gpt-oss-120b/H	46.7	49.1 (+2.4)	48.5 (+1.8) / 46.4 (-0.3)
gpt-oss-120b/L	33.6	37.6 (+4.0)	40.0 (+6.4) / 39.1 (+5.5)
gpt-oss-20b/H	46.4	46.7 (+0.3)	48.8 (+2.4) / 49.1 (+2.7)
gpt-oss-20b/L	20.0	23.9 (+3.9)	23.9 (+3.9) / 24.2 (+4.2)

hist4 より hist9 のほうが性能が向上しており，更なる履歴量の増加による改善の余地も示唆された．この性能向上の要因を探るため行動パターンを分析したところ，観測履歴の追加により前ステップと同じ行動の繰り返しが減少していた．これは観測履歴がタスク進捗の理解を助けていることを示唆する．詳細は付録 A.3 を参照されたい．

観測履歴の差分の効果 hist9 において，full 形式と diff 形式を比較すると，diff 形式でも一般に履歴なし（hist0）から性能を向上させることが分かる．また，`gpt-5.1(low)` や `o3-mini` では full 形式と同等もしくはそれ以上の効果が得られた．diff 形式では入力トークン数を 1/3 程度に抑えられるので，効率的なアプローチだといえる．

4 おわりに

本研究では，Web エージェントの観測軽量化のトレンドを再検証し，以下を明らかにした：(1) オープンソースモデルでは簡素な観測（アクセシビリティツリー）が有効だが，高性能な商用モデルではむしろ詳細な観測（HTML）が有効である (2) 商用モデルは HTML のレイアウト情報を活用できる一方，オープンソースモデルでは入力増大によりハルシネーションが増加する (3) 推論時の思考トークン量を増加させることで，詳細な観測の効果を高めることができる (4) 観測履歴の追加は多くのモデル・設定で有効であり，差分表現による効率化も可能である．

参考文献

[1] Liangbo Ning, et al. A survey of webagents: Towards next-generation ai agents for web automation with large foundation models. In **Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2**, pp. 6140–6150, 2025.

[2] Shunyu Yao, et al. React: Synergizing reasoning and acting in language models. In **The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023**, 2023.

[3] Alexandre Drouin, et al. Workarena: How capable are web agents at solving common knowledge work tasks? In **Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024**, 2024.

[4] Xiang Deng, et al. Mind2web: Towards a generalist agent for the web. In **Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023**, 2023.

[5] Ke Yang, et al. Agentoccam: A simple yet strong baseline for llm-based web agents. In **The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025**, 2025.

[6] Izzeddin Gur, et al. A real-world webagent with planning, long context understanding, and program synthesis. In **The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024**, 2024.

[7] Xing Han Lù, Zdenek Kasner, and Siva Reddy. Weblinx: Real-world website navigation with multi-turn dialogue. In **Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024**, 2024.

[8] Dongjun Lee, et al. Learning to contextualize web pages for enhanced decision making by LLM agents. In **The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025**, 2025.

[9] Saaket Agashe, et al. Agent S: an open agentic framework that uses computers like a human. In **The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025**, 2025.

[10] Xiaoqiang Wang and Bang Liu. OSCAR: operating system control via state-aware reasoning and re-planning. In **The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025**, 2025.

[11] Hanyu Lai, et al. Autowebglm: A large language model-based web navigating agent. In **Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024**, pp. 5295–5306, 2024.

[12] Yifei Wang. Beyond isolated capabilities: Bridging long cot reasoning and long-context understanding. **CoRR**, Vol. abs/2507.14849, , 2025.

[13] Dawei Zhu, et al. Chain-of-thought matters: Improving long-context language models with reasoning path supervision. **CoRR**, Vol. abs/2502.20790, , 2025.

[14] Léo Boisvert, et al. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. In **Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024**, 2024.

[15] Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. **CoRR**, Vol. abs/2507.06261, , 2025.

[16] OpenAI. gpt-oss-120b & gpt-oss-20b model card. **CoRR**, Vol. abs/2508.10925, , 2025.

[17] An Yang, et al. Qwen3 technical report. **CoRR**, Vol. abs/2505.09388, , 2025.

[18] Llama Team. The llama 3 herd of models. **CoRR**, Vol. abs/2407.21783, , 2024.

[19] Thibault Le Sellier de Chezelles, et al. The browsergym ecosystem for web agent research. **Transactions on Machine Learning Research**, 2025.

[20] Hongxin Li, et al. Autogui: Scaling GUI grounding with automatic functionality annotations from llms. In **Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025**, pp. 10323–10358, 2025.

[21] Kanzhi Cheng, et al. Seeclick: Harnessing GUI grounding for advanced visual GUI agents. In **Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024**, pp. 9313–9332, 2024.

[22] Jakub Hoscilowicz and Artur Janicki. ClickAgent: Enhancing UI location capabilities of autonomous agents. In **Proceedings of the 26th Annual Meeting of the Special Interest Group on Discourse and Dialogue**, pp. 471–476, August 2025.

[23] Hongliang He, et al. WebVoyager: Building an end-to-end web agent with large multimodal models. In **Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 6864–6890, August 2024.

[24] Yadong Lu, et al. Omniparser for pure vision based GUI agent. **CoRR**, Vol. abs/2408.00203, , 2024.

A 付録

A.1 関連研究

HTML に代表される Web ページの表現は長大であり，LLM の入力長制約や推論効率の観点から問題が生じる．このため，既存研究では観測情報の削減・選択が広く行われてきた．Deng et al. [4] は，タスク指示と HTML 要素の対応関係を cross encoder により学習し，関連度の高い要素のみを抽出して LLM に入力する枠組みを提案した．同様に，Gur et al. [6] は，専用モデルで HTML から関連要素を抽出・要約した後に LLM へ入力する二段階のパイプラインを採用している． Lü et al. [7] は dual encoder を用いて，状態表現と DOM 要素の類似度を計算して，top-k 要素を選択する手法を提案した．近年では，HTML より簡素なアクセシビリティツリー（a11y）を用いる研究も行われている．Yang et al. [5] は a11y を簡素化し，タスクに不要な要素を除外することでエージェントの性能が向上することを示した．また，Lee et al. [8] は各要素の役割を自然言語で説明することで観測を文脈化する手法を提案した．さらに，スクリーンショットのみを用いる研究 [20, 21, 22] や，スクリーンショットとテキスト観測を併用する研究 [23, 9, 24, 7] も提案されているが，やはりテキスト観測には圧縮・選択された表現が用いられている．

現在の観測に加え，過去の観測や行動の履歴を利用することも性能向上に有効である．しかし全履歴の保持は現実的でなく，履歴の情報量を制御する工夫が行われてきた． Lai et al. [11] は，行動ログと局所的な HTML 断片のみを履歴として保存することで，履歴の冗長性を抑えている．また，Yang et al. [5] は，現在実行中のサブタスクに関連するステップの観測のみを保持するようにしている．

まとめ 既存研究では，単一ステップ観測および観測履歴のいずれにおいても，情報量を削減する設計が主流であった．しかし，長文処理に優れたモデルや思考トークン量が多い設定でも軽量化が有効かについては，十分に検証されていない．

A.2 グラウンディングエラーの分類

Web エージェントが出力した行動が環境に正しく受理されなかった，または実行前の前提条件チェックで失敗したエラーを**グラウンディングエラー**と定義する．一方，行動自体は正しく構成されていたが，ネットワーク障害やタイムアウト等の環境側の問題により失敗したエラーは分析対象から除外した．グラウンディングエラーは，エラーメッセージに含まれる文字列パターンに基づき分類した．分類の説明を表 3 に示す．

表 3 グラウンディングエラーの分類

種別	説明
API 形式エラー：BrowserGym API の呼び出し形式が誤り	
Scroll	`scroll()` 関数に不正な引数を渡した
Click	`click()` 関数に不正な引数を渡した
Select	`select_option()` 関数に不正な引数を渡した
要素可用性エラー：指定した要素の特定または状態確認で発生	
Not Found	指定した要素が DOM 上に存在しない
Not Visible	指定要素が CSS により非表示
Not Enabled	指定要素が `disabled` 属性等により無効化
Wrong Type	指定要素では，指定した操作を受け付けない
Option Not Found	指定要素には，`select_option()` で指定したオプション値が存在しない
操作実行時エラー：操作実行時に発生	
Intercepted	別要素によりクリック操作がブロック
Invalid URL	`goto()` で指定された URL が無効

A.3 行動反復率の分析

§3.4 で示した観測履歴の効果を分析するため，エージェントが前ステップと同一の行動を繰り返した割合（反復率）を計測した． 観測履歴により過去の行動の結果を把握できれば，タスク進捗の理解が促進され，同一行動の繰り返しは減少すると考えられる． 図 3 にタスク成功率と反復率の関係を示す．反復率が低いほど成功率は高く，観測履歴の追加により反復率は減少する傾向が確認された． 特に思考トークン量が少ない場合，観測履歴の追加による反復率の減少効果が顕著であった． この結果は，観測履歴がタスク進捗の理解を助けることで性能向上に寄与していることを示唆する．

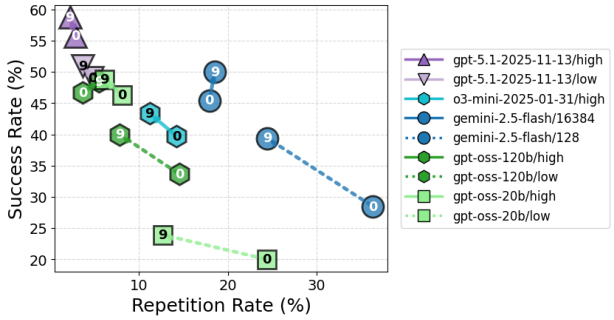

図 3 行動の反復率と成功率の関係（WorkArena L1）．x 軸は前ステップと同一の行動を繰り返した割合，y 軸はタスクの成功率．各データ点の数字は入力に含める観測履歴の長さを表す．